Name: Anushka Harshavadan Nevgi
Experiment: 10. Implement a program to demonstrate Control Flow Graph.
Class: TY CSE A

```
#include <iostream>
#include <vector>
#include <string>
#include <unordered_map>
#include <set>

using namespace std;

// Class representing a Basic Block
class BasicBlock {
public:
    int id;                 // Block ID
    string label;           // Label or name of the basic block
    vector<string> code;    // Code inside the block

    // Default constructor (needed for unordered_map)
    BasicBlock() : id(0), label("") {}

    // Parameterized constructor
    BasicBlock(int id, const string& label) : id(id), label(label) {}

    // Add a line of code to the basic block
    void addCode(const string& line) {
        code.push_back(line);
    }

    // **Fix: Marked display() as 'const' to allow calling on const objects**
    void display() const {
        cout << "Basic Block " << id << " (" << label << "):\n";
        for (const string& line : code) {
            cout << "  " << line << endl;
        }
    }
};

// Class representing the Control Flow Graph (CFG)
class CFG {
public:
    unordered_map<int, BasicBlock> blocks;
// Map of block ID to Basic Block
    unordered_map<int, set<int>> edges;     // Map of block ID to a set of successor block IDs

    // Add a basic block to the CFG
    void addBlock(const BasicBlock& block) {
        blocks[block.id] = block;
    }

    // Add a directed edge between two basic blocks
    void addEdge(int fromBlockId, int toBlockId) {
        edges[fromBlockId].insert(toBlockId);
    }

    // Display the Control Flow Graph
    void display() const {
        for (const auto& pair : blocks) {
            pair.second.display();
            if (edges.find(pair.first) != edges.end()) {
                cout << "  Successors: ";
                for (int succ : edges.at(pair.first)) {
                    cout << succ << " ";
                }
                cout << endl;
            }
        }
    }
};

// Main function to construct and display the CFG
int main() {
```

```
    // Create Basic Blocks
    BasicBlock block1(1, "Entry");
    block1.addCode("int a = 0;");

    BasicBlock block2(2, "Condition");
    block2.addCode("if (a > 0) { ... } else { ...
}");

    BasicBlock block3(3, "Increment");
    block3.addCode("a = a + 1;");

    BasicBlock block4(4, "Decrement");
    block4.addCode("a = a - 1;");

    BasicBlock block5(5, "Return");
    block5.addCode("return 0;");

    // Create the Control Flow Graph (CFG)
    CFG cfg;

    // Add basic blocks to the CFG
    cfg.addBlock(block1);
    cfg.addBlock(block2);
    cfg.addBlock(block3);
    cfg.addBlock(block4);
    cfg.addBlock(block5);

    // Add edges to represent control flow
    cfg.addEdge(1, 2); // Entry -> Condition
    cfg.addEdge(2, 3); // Condition ->
Increment
    cfg.addEdge(2, 4); // Condition ->
Decrement
    cfg.addEdge(3, 5); // Increment -> Return
    cfg.addEdge(4, 5); // Decrement -> Return

    // Display the Control Flow Graph
    cout << "Control Flow Graph:\n";
    cfg.display();

    return 0;
}
```

```
Control Flow Graph:
Basic Block 5 (Return):
  return 0;
Basic Block 4 (Decrement):
  a = a - 1;
  Successors: 5
Basic Block 3 (Increment):
  a = a + 1;
  Successors: 5
Basic Block 2 (Condition):
  if (a > 0) { ... } else { ... }
  Successors: 3 4
Basic Block 1 (Entry):
  int a = 0;
  Successors: 2
```